ECE 2300
Digital Logic & Computer Organization

Spring 2025

Single Cycle Microprocessor

Cornell University

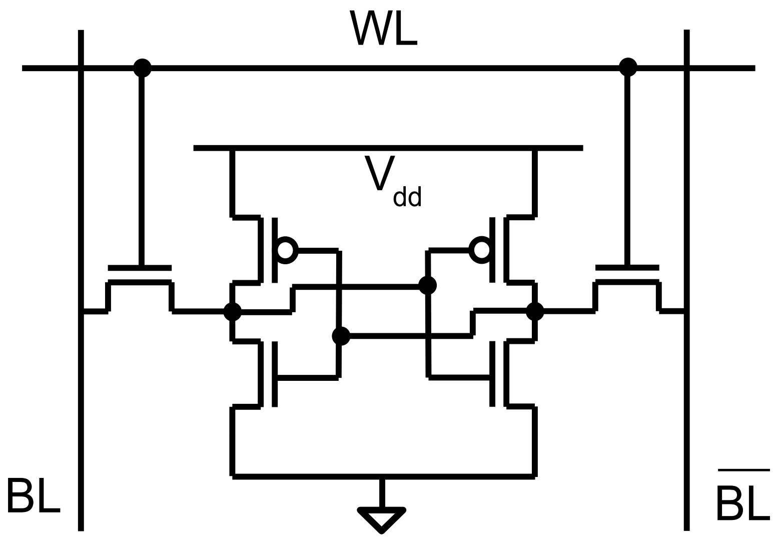
Announcements

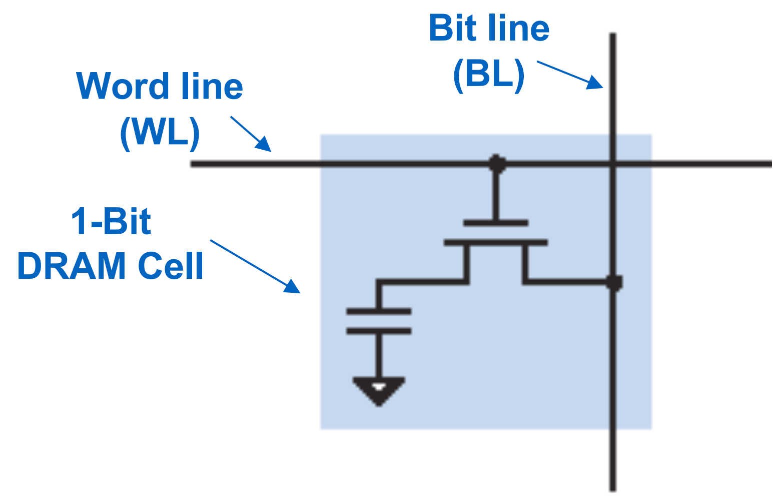
- HW 5 due tomorrow

- HW 6 will be posted today
  – Covers Lecture 16

- Prelim 2: Thursday April 10, 7:30pm, 90mins, GSH G76
  – Coverage: Lectures 8~16
    - FSMs, timing analysis, binary arithmetic, memories, single-cycle microprocessor (no Verilog)
  – A sample exam will be posted on CMS next week
  – TA-led review will be scheduled on Monday 4/7
  – Email the instructor asap if you have a conflict

SRAM vs. DRAM





- SRAM: usually on the same chip with CPU

  (+) Fast access

  (−) Relatively high area & cost per bit (6T)

- DRAM: typically off-chip & used for main memory

  (+) Single transistor bit cell (1T1C)

  - Lower area & lower cost per bit

  (−) Slow: need periodic refresh

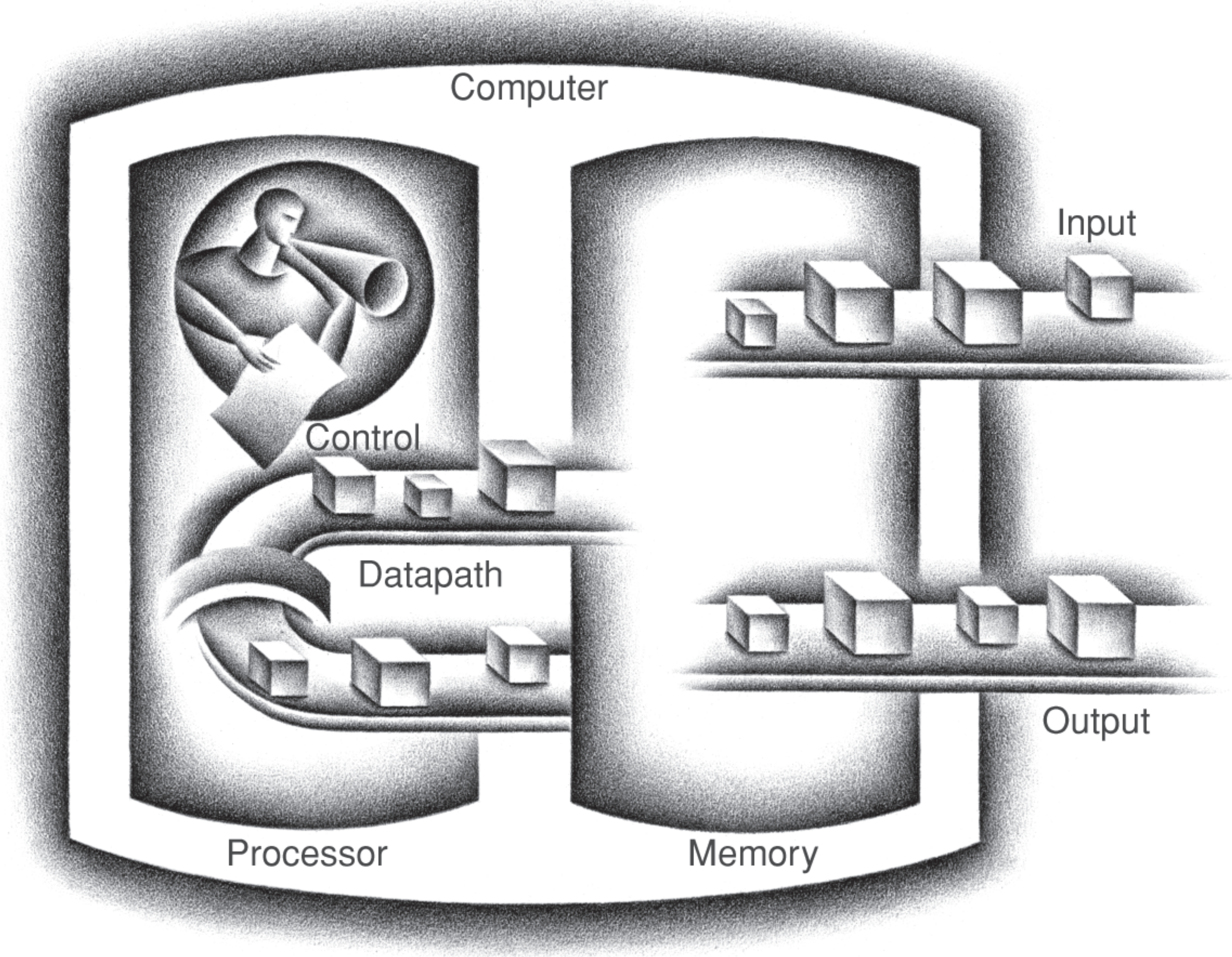
SRAM typically ranges from KB to MB, while DRAM is usually in GB.

Computer Memory Explained (by analogy)

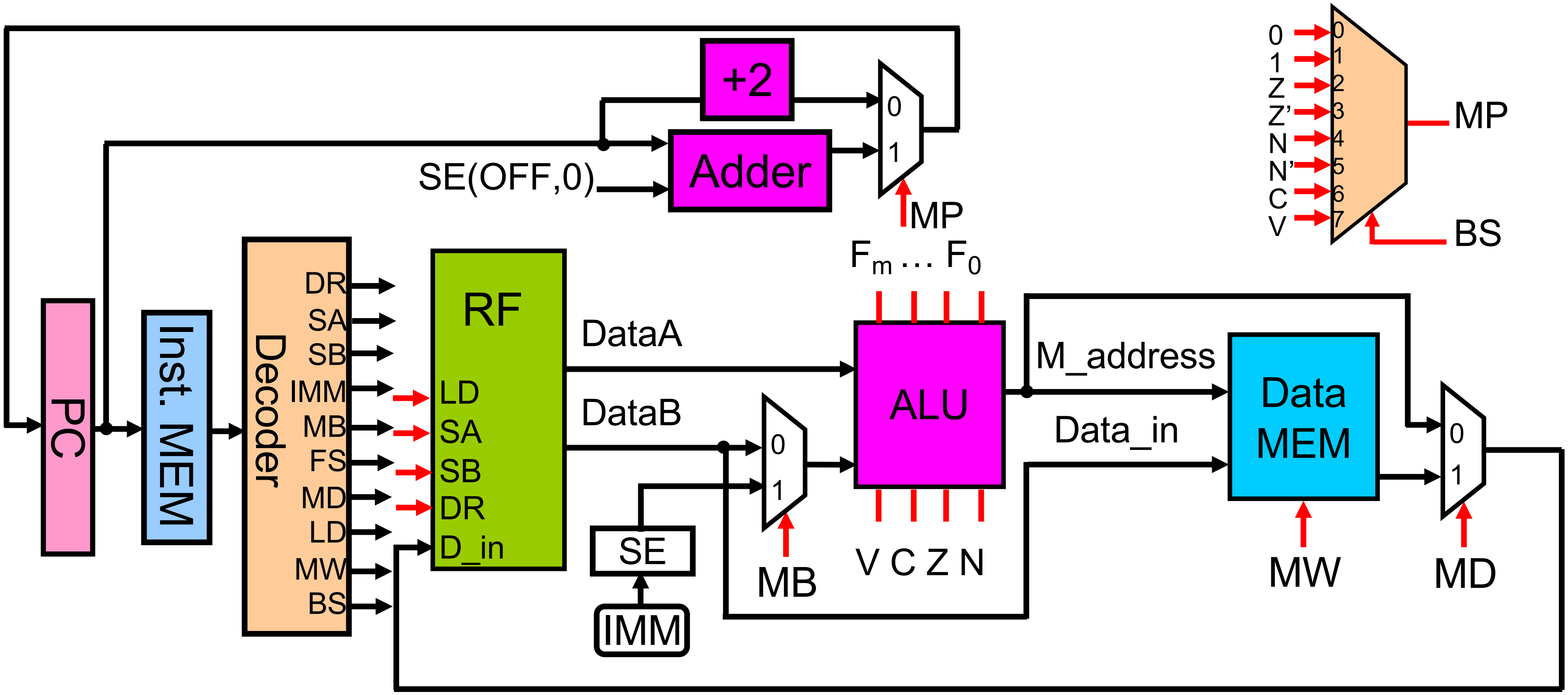
Course Roadmap (Part 1)

- Boolean algebra
- Combinational logic and minimization
- Logic functions
- CMOS gates
- Binary arithmetic and ALUs
- Latches and flip-flops
- Counters
- Verilog
- Finite state machines
- Hazards, timing, clocking
- Memories

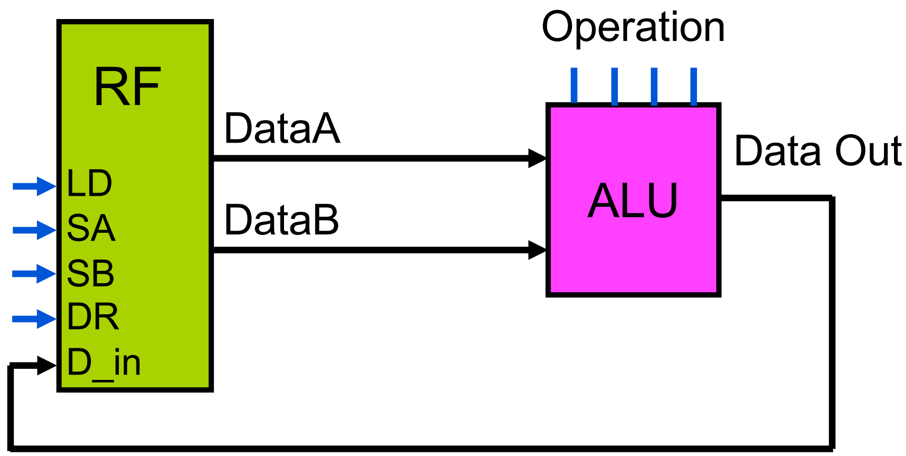
Part 2: Computer Organization



Let's Build a Microprocessor!



The Compute Core

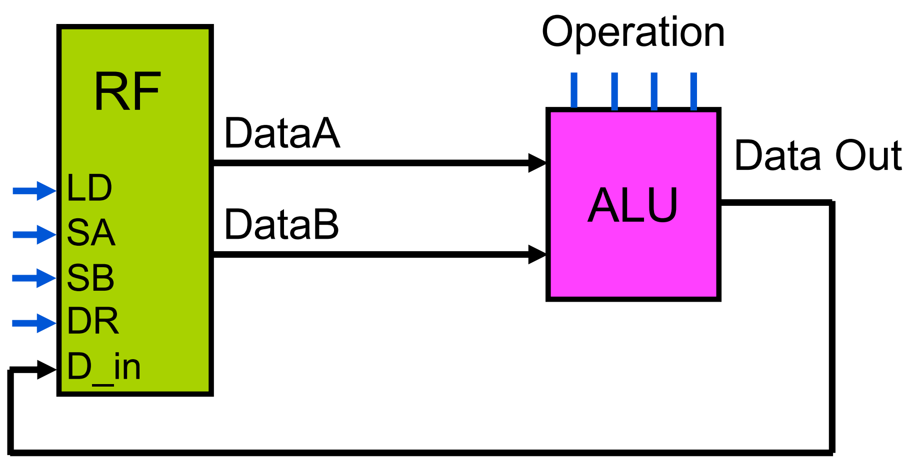


- **The processor has internal memory called the register file (RF), built with DFFs or SRAM, which are much faster to access than DRAM, but also a lot more expensive[1]**

- **The ALU reads data from the RF, performs computations, and write back the result**

[1] The RF is small, so data must be moved between main memory and RF using explicit load/store instructions (covered in later slides).

The Basic Processing Cycle



- **Read data from two registers**
- **Perform an operation**
- **Place the result into a register**

- **All three steps performed in 1 clock cycle**

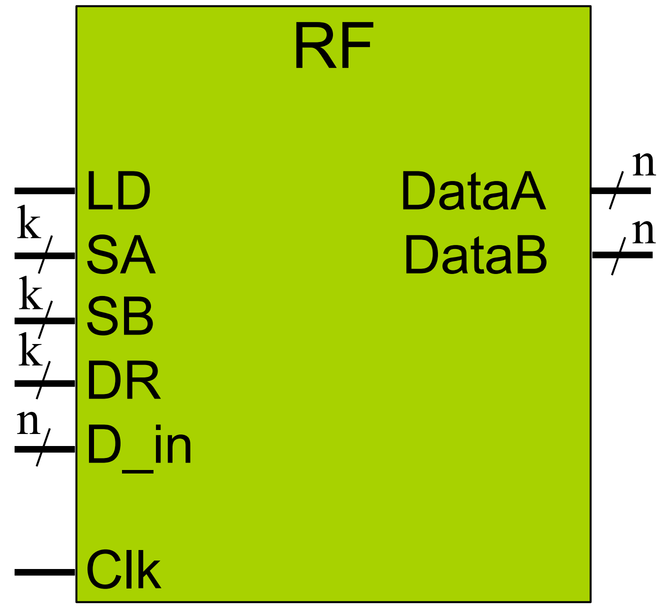
Register File (RF)

- **Collection of 2^k n-bit registers**
- **Data outputs (two read ports)**

  DataA – Output data A

  DataB – Output data B
- **Data inputs (one write port)**

  D_in – Input data



- **Control inputs**

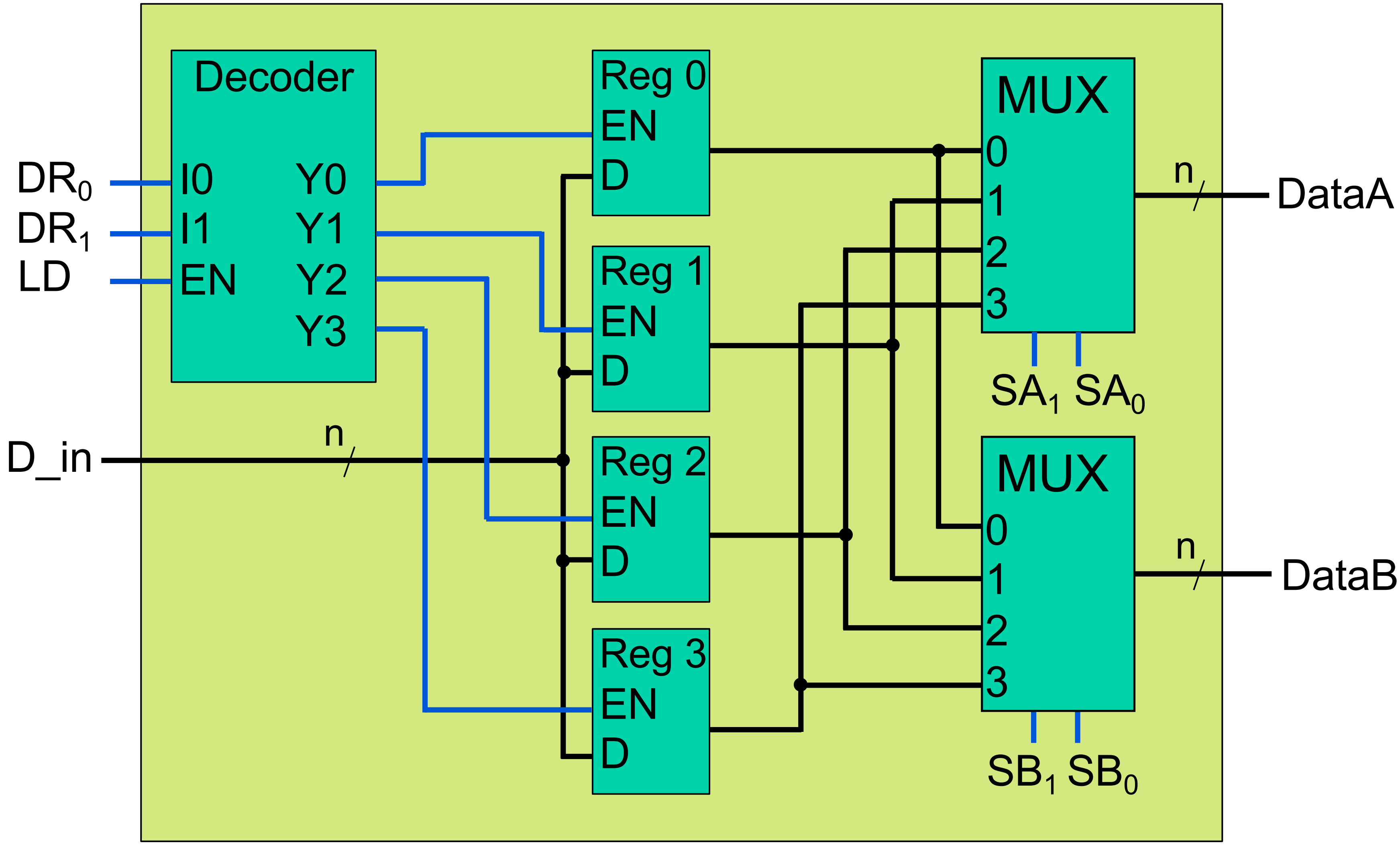
  SA – Address (or index) of source register A

  SB – Address of source register B

  DR – Address of destination register

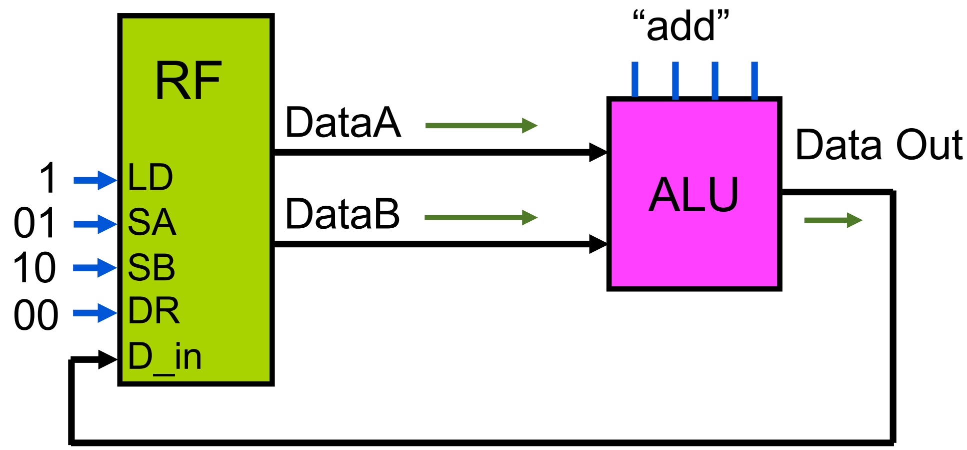
  LD (i.e., write enable) – Load D_in into destination register DR

Example RF Organization



Example with 4 registers. Typically have 32 or more.

Instruction Execution



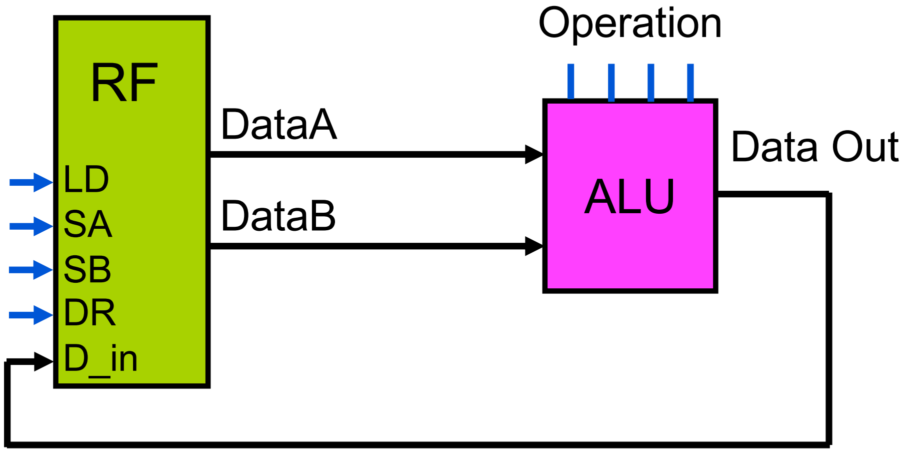
Example: **ADD R0, R1, R2** **// R0 <= R1 + R2**

operation

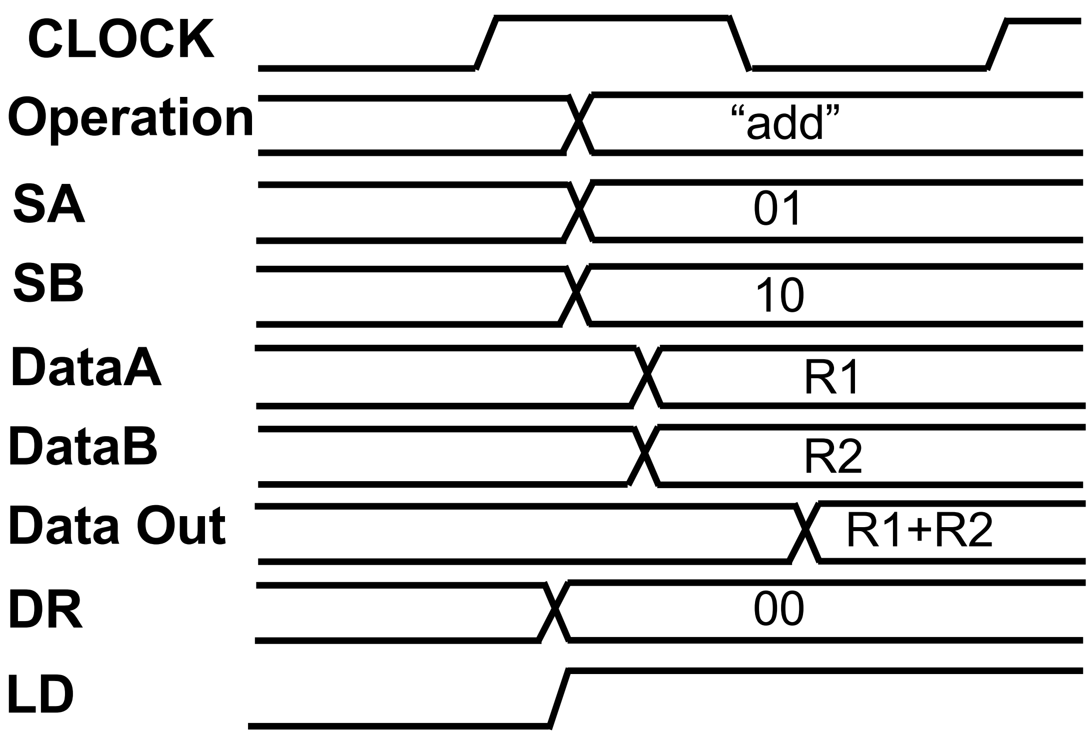
destination register
(write to Reg 0)

source registers
(read from Reg 1 & Reg 2)

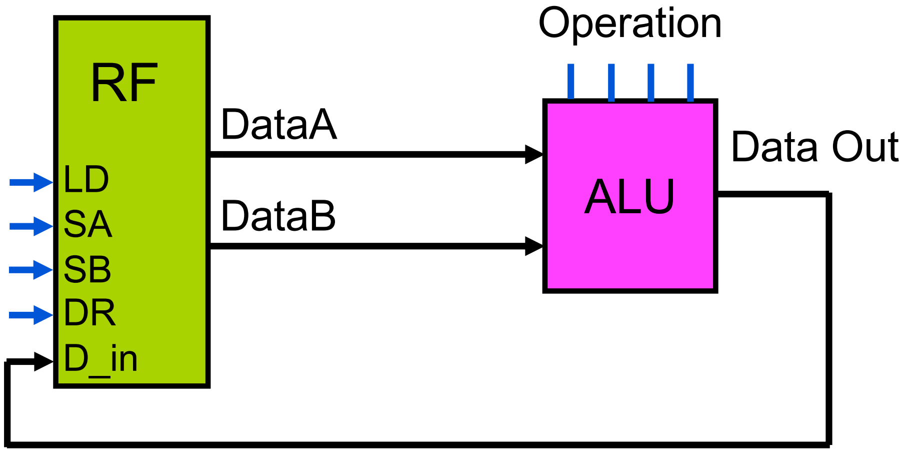
Instruction Execution



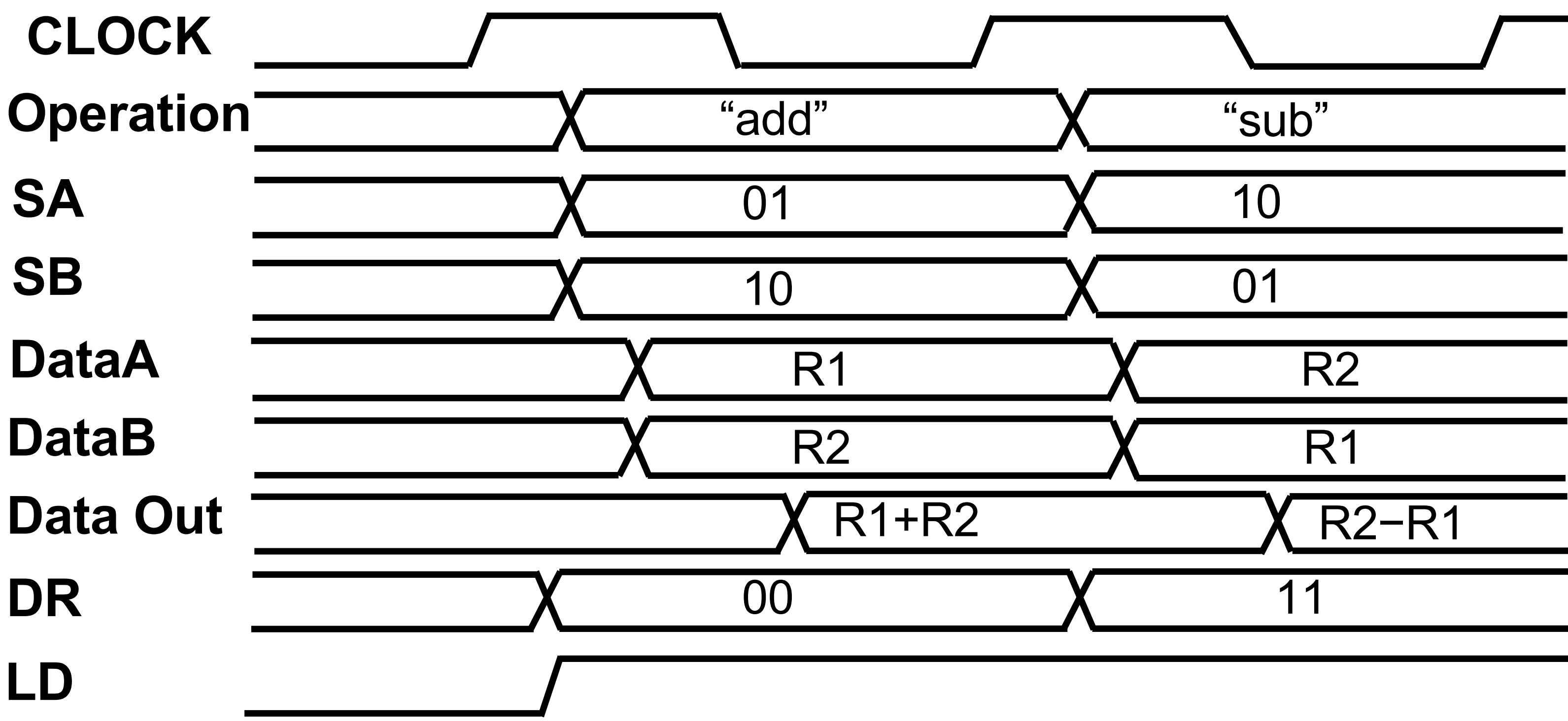
ADD R0, R1, R2



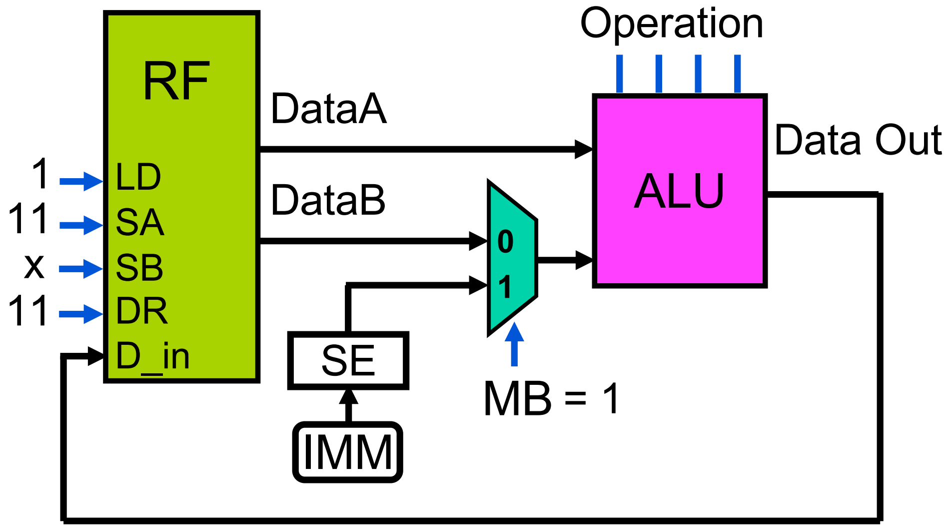
Instruction Execution



ADD R0, R1, R2
SUB R3, R2, R1



Operations With Constants



ADDI R3, R3, 1

// R3 <= R3 + 1

- **Constants are called *immediate values* (IMM)**
- **Sign extend (SE) IMM to the width of DataA to perform correct two's complement operation**
  - **Why? not enough bits in instruction (explained later)**
  - **Assume IMM is 4 bits and DataA is 8 bits wide**

0001 ⟶ 00000001

1110 ⟶ 11111110

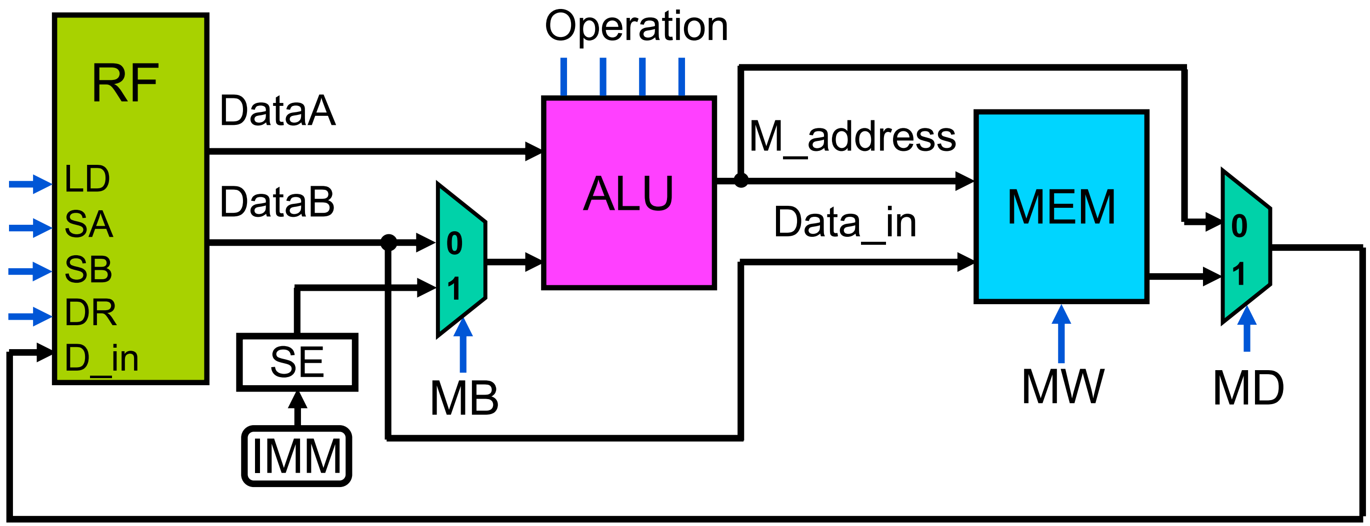
Sign Extension

- Replicate the MSB (sign bit)

<u>**4-bit**</u>			<u>**8-bit**</u>	
0100	(4)	→	**00000100**	(still 4)
1100	(-4)	→	**11111100**	(still -4)

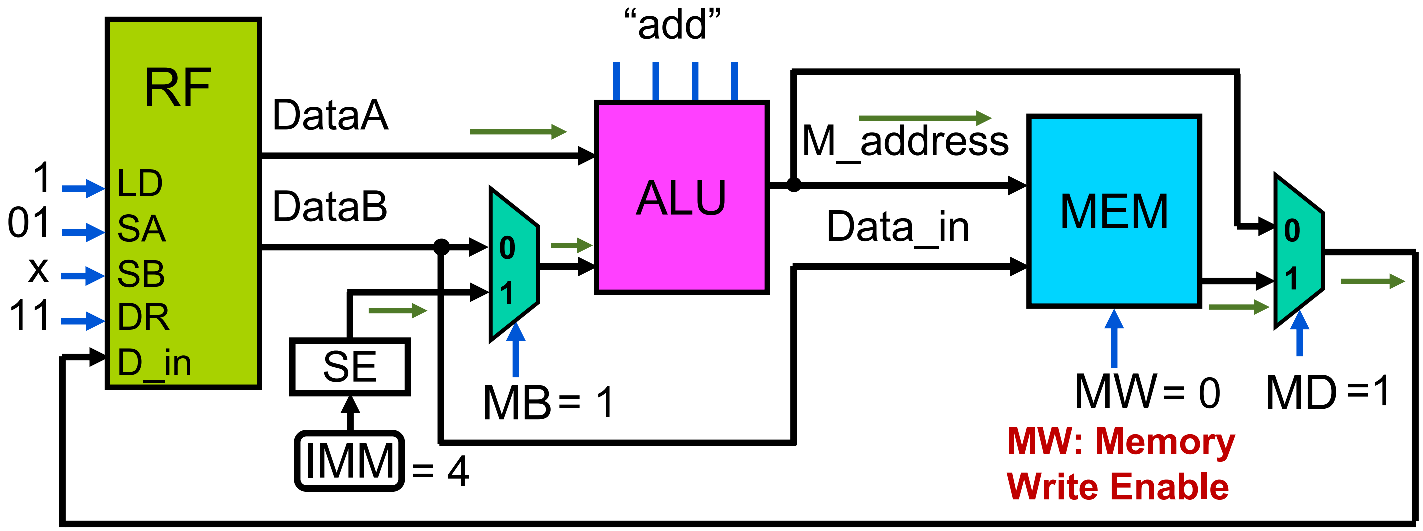
- Necessary for aligning two's complement numbers of different lengths before a fixed-size arithmetic operation

Reading and Writing Memory



- Most data are held in main memory (MEM)
- Must be moved into registers for ALU to operate on them
- Data will also move out of registers into memory
  - To make room for other data
  - Or to later move it to permanent storage (e.g., disk)

Reading from Memory ("Load")



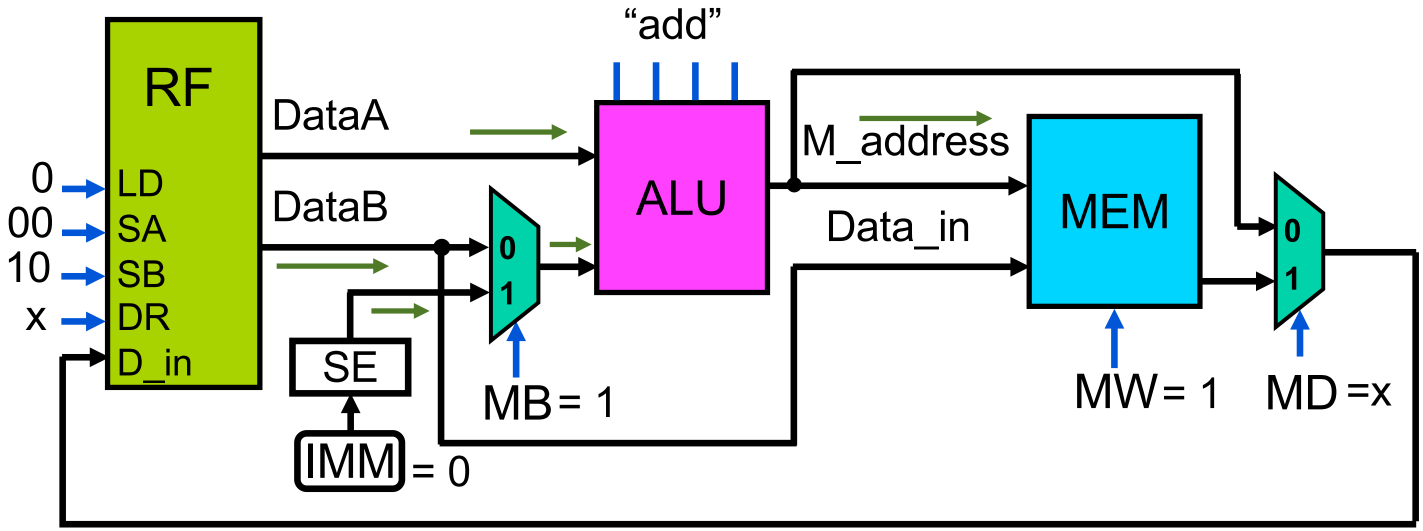
Example: LOAD R3, 4(R1) // R3 <= M[R1 + 4]

M means "Main Memory"

Step 1: Form the memory address by adding R1 (*base address*) with the immediate 4 (*offset*)

Step 2: Read the data at that address in RAM (M[R1+4]) and place it in R3

Writing to Memory ("Store")



Example: **STORE R2, 0(R0)** **// M[R0] <= R2**

Step 1: Form the memory address by adding R0 (base) with the immediate 0 (offset)

Step 2: Write the data in R2 into the RAM at that address (M[R0+0], i.e., M[R0])

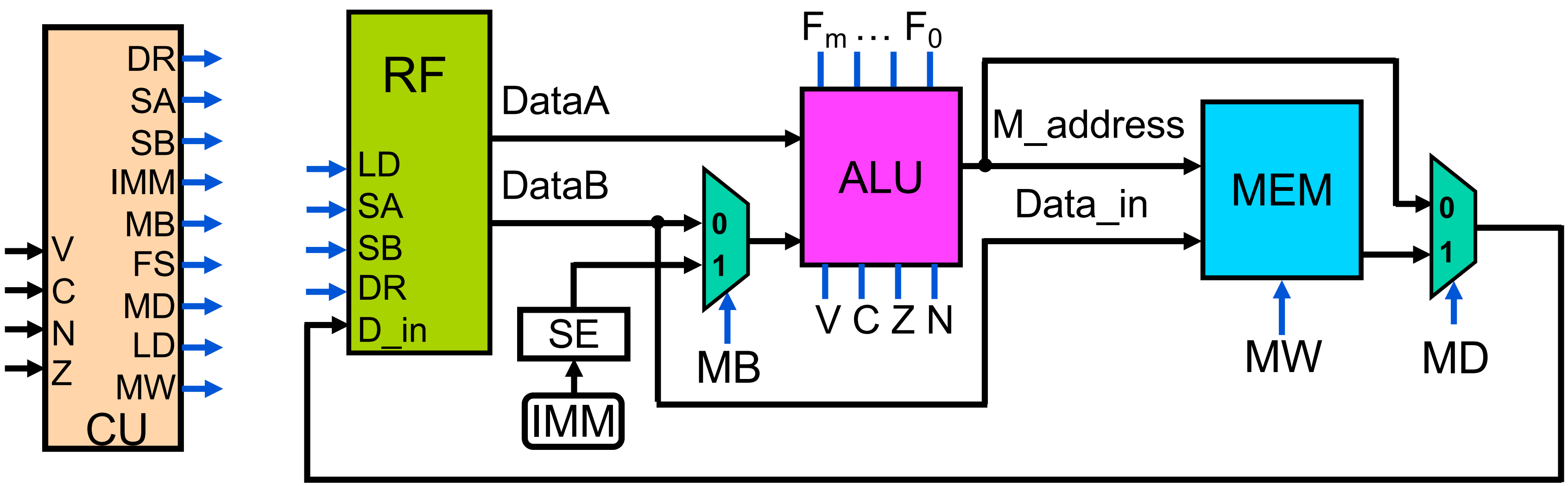
Control Unit (CU)

- Regulates the interaction between data and operations on data (i.e., datapath)

- Series of *control words* control the datapath to perform a sequence of operations

- The sequence of operations performed by the CU may be affected by the ALU *Condition Codes*
  – Z: Zero
  – N: Negative
  – Also V: Overflow and C: Carry out

Datapath + Control Unit



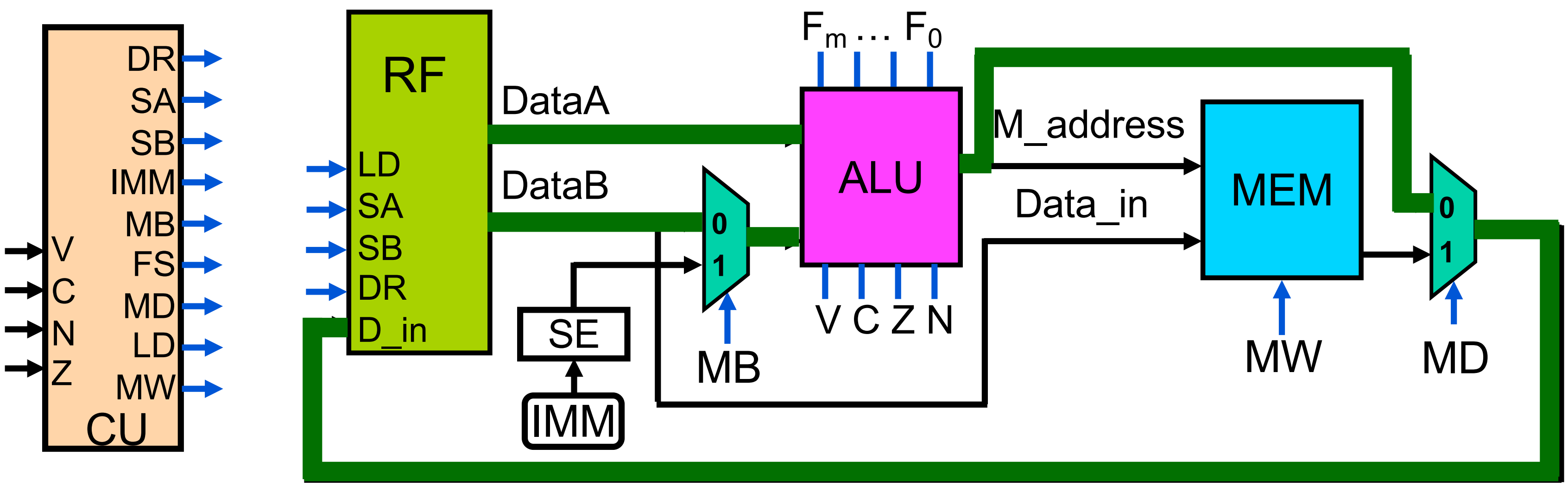
Control Word → **(output of CU)**

DR	SA	SB	IMM	MB	FS	MD	LD	MW

- DR, SA, SB: k-bit register addresses
- IMM: immediate value
- MB: register/immediate select
- FS: function select
- MD: ALU/memory select
- LD: load register
- MW: memory write

Sequence of Operations

Assuming 8 registers in the RF

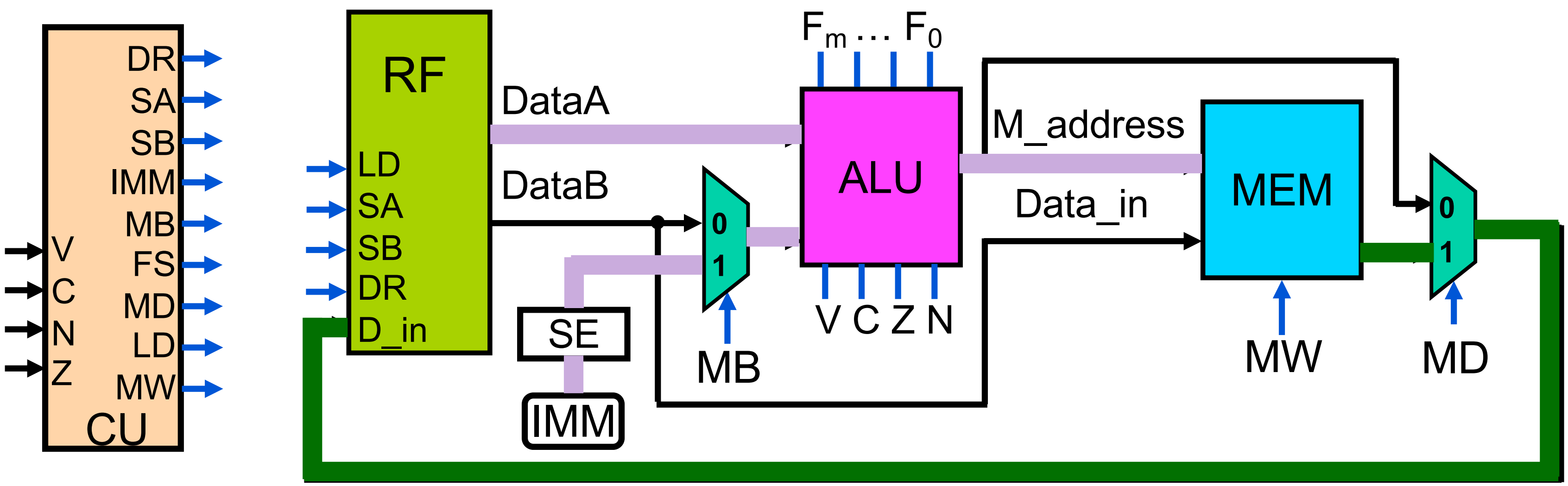


DR	SA	SB	IMM	MB	FS	MD	LD	MW
010	**000**	**001**	**x**	**0**	**ADD**	**0**	**1**	**0**

R2 <= R0 + R1

Sequence of Operations

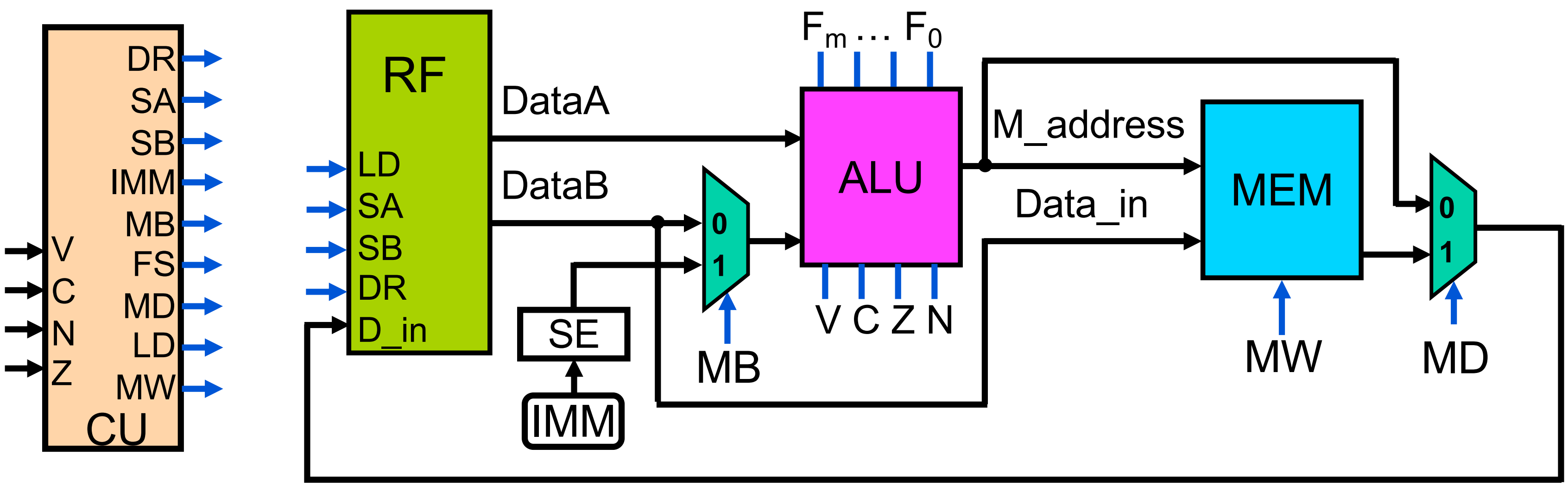
Assuming 8 registers in the RF



	DR	SA	SB	IMM	MB	FS	MD	LD	MW
R2 <= R0 + R1	010	000	001	x	0	ADD	0	1	0
R1 <= M[R2]	**001**	**010**	**x**	**0**	**1**	**ADD**	**1**	**1**	**0**

Sequence of Operations

Assuming 8 registers in the RF



	DR	SA	SB	IMM	MB	FS	MD	LD	MW
R2 <= R0 + R1	010	000	001	x	0	ADD	0	1	0
R1 <= M[R2]	001	010	x	0	1	ADD	1	1	0
	x	010	000	0	1	ADD	x	0	1
R3 <= R1 – 3	011	001		3		SUB			

Next Class

More Single Cycle Microprocessor

(H&H 7.3)